Variational Treatment of Probabilistic Directed Graphical Models

Satish Kumar

July 27, 2018

LEAP Lab, IISC Bengaluru

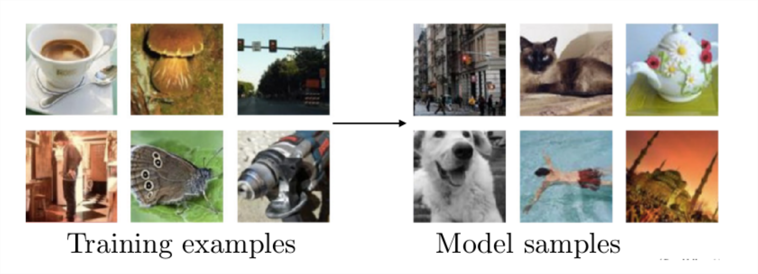
Table of contents

Generative modeling

Density Estimation

Figure 1: Some generative models perform density estimation. These models take a training set of examples drawn from an unknown data-generating distribution p_{data} and return an estimate of that distribution. The estimate p_{model} can be evaluated for a particular value of $\boldsymbol{x}$ to obtain an estimate $p_{\text{model}}(\boldsymbol{x})$ of the true density $p_{\text{model}}(\boldsymbol{x})$. This figure illustrates the process for a collection of samples of one-dimensional data and a Gaussian model.

An Ideal Generative model



Why study Generative modeling?

- Training and sampling from generative models is an excellent test of our ability to represent and manipulate high-dimensional probability distributions.
- Generative models can be trained with missing data and can provide predictions on inputs that are missing data.
- It enable machine learning to work with multi-modal outputs
- Realistic generation of samples from some distribution.

Evidence Lower Bound(ELBO)

EM:A Latent Variable View

Consider a probabilistic model:Observed variables X and latent variables Z.Our goal is to minimize the likelihood function given by

$$p(\mathbf{X}|\boldsymbol{\theta}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$$

ELBO:Evidence Lower Bound

- We introduce a distribution q(z) defined over latent variables,and for any choice of q(z) the following decomposition holds

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{KL}(q\|p)$$

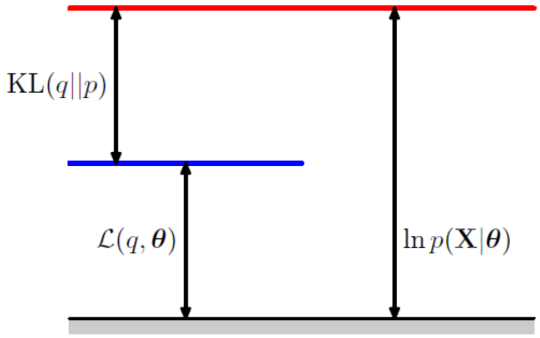
where we have defined

$$\begin{aligned}
\mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \\
\mathrm{KL}(q\|p) &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\}.
\end{aligned}$$

- using the product rule of probability

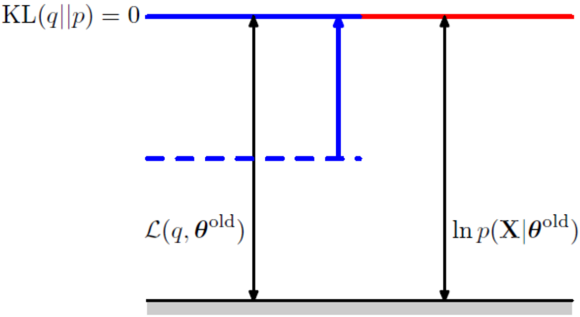
$$\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \ln p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}) + \ln p(\mathbf{X}|\boldsymbol{\theta})$$

Illustration of ELBO



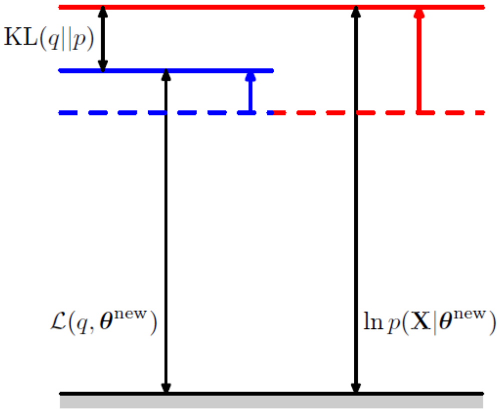
E-step

Illustration of the E step of the EM algorithm. The q distribution is set equal to the posterior distribution for the current parameter values $\boldsymbol{\theta}^{\text{old}}$, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.



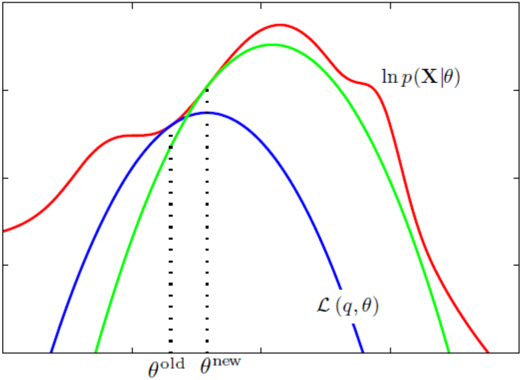
M-step

Illustration of the M step of the EM algorithm. The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to the parameter vector $\boldsymbol{\theta}$ to give a revised value $\boldsymbol{\theta}^{\text{new}}$. Because the KL divergence is nonnegative, this causes the log likelihood $\ln p(\mathbf{X}|\boldsymbol{\theta})$ to increase by at least as much as the lower bound does.



Animation of Lower Bound

The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.



Variational Inference in Deep Learning

Some preliminaries

- We assume the observed variable $\boldsymbol{x}$ is a random sample from an unknown underlying process where the distribution $p^*(\boldsymbol{x})$ is unknown
- we approximate this process with a chosen model $p_\theta(\boldsymbol{x})$ with parameter θ

$$\boldsymbol{x} \sim p_\theta(\boldsymbol{x})$$

- Learnig is the process of searching the θ such that,for any observed variable $\boldsymbol{x}$

$$p_\theta(\boldsymbol{x}) \approx p^*(\boldsymbol{x})$$

- we wish $p_\theta(\boldsymbol{x})$ to be sufficiently flexible to be able to adapt to the data
- Often, such as in case of classification or regression problems, we are not interested in learning an unconditional model $p_\theta(\boldsymbol{x})$, but a conditional model $p_\theta(\boldsymbol{y}|\boldsymbol{x})$
- that approximates the underlying conditional distribution $p^*(\boldsymbol{y}|\boldsymbol{x})$

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) \approx p^*(\boldsymbol{y}|\boldsymbol{x})$$

Parameterizing Conditional distributions with Neural Nets

- We parameterize conditional distributions with neural networks
- In image classification, neural networks parameterize a categorical distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ over a class label $\boldsymbol{y}$, conditioned on an image $\boldsymbol{x}$.

$$\mathbf{p} = \mathrm{NeuralNet}(\mathbf{x})$$
$$p_{\boldsymbol{\theta}}(y|\mathbf{x}) = \mathrm{Categorical}(y; \mathbf{p})$$

- where for all $p_i \in \boldsymbol{p}$
- and the last operation of the neural net is a *softmax*() function such that $\sum_i p_i = 1$

In the case of a Gaussian MLP as encoder or decoder, we let the encoder or decoder be a multivariate Gaussian with a diagonal covariance structure:

$$
\begin{aligned}
\log p(\mathbf{x}|\mathbf{z}) &= \log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \\
\text{where } \boldsymbol{\mu} &= \mathbf{W}_4\mathbf{h} + \mathbf{b}_4 \\
\log \boldsymbol{\sigma}^2 &= \mathbf{W}_5\mathbf{h} + \mathbf{b}_5 \\
\mathbf{h} &= \tanh(\mathbf{W}_3\mathbf{z} + \mathbf{b}_3)
\end{aligned}
$$

where $\{\mathbf{W}_3, \mathbf{W}_4, \mathbf{W}_5, \mathbf{b}_3, \mathbf{b}_4, \mathbf{b}_5\}$ are the weights and biases of the MLP and part of $\boldsymbol{\theta}$ when used as decoder. Note that when this network is used as an encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, then $\mathbf{z}$ and $\mathbf{x}$ are swapped, and the weights and biases are variational parameters $\boldsymbol{\phi}$.

Learning in Fully Observed models with neural nets

- **Dataset**:We often collect a dataset $\boldsymbol{D}$ consisting of $N \geq 1$ datapoints:

$$\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\} \equiv \{\mathbf{x}^{(i)}\}_{i=1}^{N} \equiv \mathbf{x}^{(1:N)}$$

- Under the i.i.d. assumption, the probability of the datapoints given the parameters factorizes as a product of individual datapoint probabilities.
- **Maximum Likelihood and Minibatch SGD**

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

- Using automatic differentiation tools, we can efficiently compute gradients of this objective, and use such gradient to find the local optimum of the ML objective.
- We can opt either Stochastic Gradient or Batch Gradient Methods

Learning and Inference in Deep latent variable models

- **Latent Variables, z** are variables that are part of the model, but which we dont observe, and are therefore **not part of the dataset**.
- In case of unconditional modeling of observed variable $\boldsymbol{x}$, the directed graphical model would then represent a joint distribution $p_\theta(\boldsymbol{x},\boldsymbol{z})$ over both the observed variables $\boldsymbol{x}$ and the latent variables $\boldsymbol{z}$.
- The marginal distribution over the observed variables $p_\theta(\boldsymbol{x})$, is given by:

$$p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x},\boldsymbol{z}) d\boldsymbol{z}$$

- This is also called the (single datapoint) **marginal likelihood** or the **model evidence**, when taking as a function of $\boldsymbol{\theta}$.

Deep Latent Variable Models(DLVMs)

- We use the term deep latent variable model (DLVM) to denote a latent variable model $p_\theta(\boldsymbol{x},\boldsymbol{z})$ whose distributions are parameterized by neural networks
- Such a model can be conditioned on some context, like $p_\theta(\boldsymbol{x},\boldsymbol{z}|\boldsymbol{y})$
- The simplest, and most common, graphical model with latent variables is one that is specified as factorization with the following structure:

$$p_\theta(\boldsymbol{x},\boldsymbol{z})=p_\theta(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})$$

- The distribution $p(\boldsymbol{z})$ is often called the prior distribution over $\boldsymbol{z}$

Example: DLVM for multivariate Bernoulli data

- For binary data **x**, with a spherical Gaussian latent space, and a factorized Bernoulli observation model:

$$
\begin{aligned}
p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; 0, \mathbf{I}) \\
\mathbf{p} &= \text{DecoderNeuralNet}_{\theta}(\mathbf{z}) \\
\log p(\mathbf{x}|\mathbf{z}) &= \sum_{j=1}^{D} \log p(x_j|\mathbf{z}) = \sum_{j=1}^{D} \log \text{Bernoulli}(x_j; p_j) \\
&= \sum_{j=1}^{D} x_j \log p_j + (1 - x_j) \log(1 - p_j)
\end{aligned}
$$

- where $\forall p_j \;\; \mathbf{p} : 0 \leq p_j \leq 1$ (e.g. implemented through a sigmoid nonlinearity as the last layer of the *DecoderNeuralNet*(.)), where D is the dimensionality of $\boldsymbol{x}$,
- *Bernoulli*$(.; p)$ is the probability mass function (PMF) of the Bernoulli distribution.

Intractabilities

- The main difficulty of maximum likelihood learning in DLVMs is that the marginal probability of data under the model is typically **intractable**.
- This is due to the integral in equation for computing the marginal likelihood (or modelevidence)
- Note that the joint distribution $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ is efficient to compute, and that the densities are related through the basic identity:

$$p_\theta(\boldsymbol{z}|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}, \boldsymbol{z})}{p_\theta(\boldsymbol{x})}$$

- Since $p_\theta(\boldsymbol{x}, \boldsymbol{z})$ is tractable to compute
- a tractable marginal likelihood $p_\theta(\boldsymbol{x})$ leads to a tractable posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, and vice versa
- But both are intractable
- Approximate inference techniques allow us to approximate the posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$ and the marginal likelihood $p_\theta(\boldsymbol{x})$ in DLVMs.

Variational Autoencoder

Encoder or approximate posterior

- Let $p_\theta(\boldsymbol{x},\boldsymbol{z})$ be a latent-variable model with observed variables x and latent variables $\boldsymbol{z}$
- To turn a DLVMs intractable posterior inference and learning problems into tractable problems, we introduce a parametric inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
- This model is also called an encoder.
- With $\boldsymbol{\phi}$ we indicate the parameters of this inference model, also called the variational parameters.
- We optimize the variational parameters such that:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{z}|\boldsymbol{x})$$

- As we will explain, this approximation to the posterior help us optimize the marginal likelihood.

- Like a DLVM, the inference model can be (almost) any directed graphical model:

$$q_\phi(\mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}_1, ..., \mathbf{z}_M|\mathbf{x}) = \prod_{j=1}^{M} q_\phi(\mathbf{z}_j|Pa(\mathbf{z}_j), \mathbf{x})$$

- similar to a DLVM, the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ can be parameterized using deep neural networks
- In this case, the variational parameters $\boldsymbol{\phi}$ include the weights and biases of the neural network

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_\phi(\mathbf{x})$$
$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}))$$

- we use a single encoder neural network to perform posterior inference over all of the datapoints in our dataset

Evidence lower bound for VAE

- The **optimization objective** of the variational autoencoder, like in other variational methods, is the evidence lower bound
- Also called Variational lower bound
- For any choice of inference model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, including the choice of variational parameters , we have:

$$
\begin{aligned}
\log p_\theta(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x})\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\theta,\phi}(\mathbf{x}) \\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))}
\end{aligned}
$$

- KL divergence between $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and $p_\theta(\boldsymbol{z}|\boldsymbol{x})$,which is non-negative:

$$D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \geq 0$$

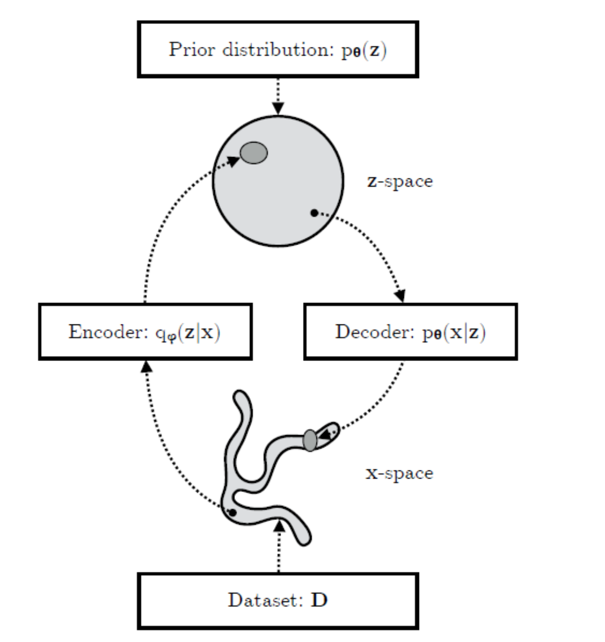
- and zero if, and only if, $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ equals the true posterior distribution
- The first term in the
  variational lower bound, also called the evidence lower bound (ELBO):

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\theta}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})\right]$$

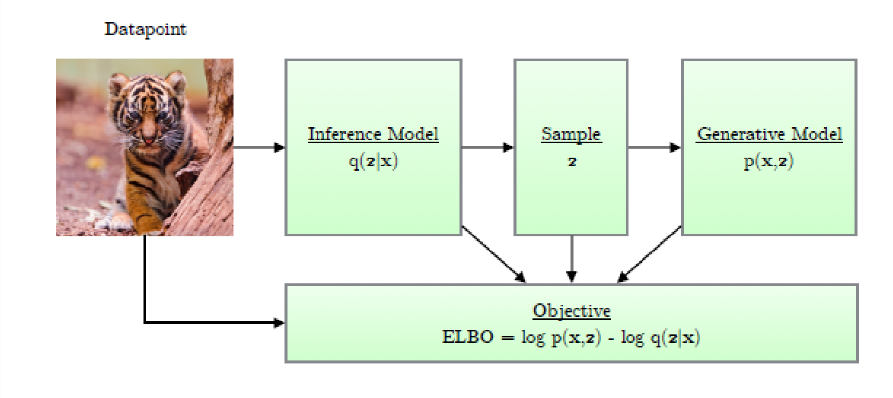
- Due to the non-negativity of the *KLDivergence*, the ELBO is a lower bound on the log-likelihood of the data.

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \log p_{\theta}(\mathbf{x}) - D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}|\mathbf{x})) \\ &\leq \log p_{\theta}(\mathbf{x})\end{aligned}$$

VAE in a nutshell



Control flow in VAE



A Double-Edged Sword

- Maximization of the ELBO $L_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x})$ w.r.t. the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, will concurrently optimize the two things we care about:
- It will approximately maximize the marginal likelihood $p_\theta(\boldsymbol{x})$. This means that our generative model will become better.
- It will minimize the KL divergence of the approximation $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ from the true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$, so $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ becomes better.

Stochastic gradient-based optimization of the ELBO

- An important property of the ELBO, is that it allows joint optimization w.r.t. all parameters ($\boldsymbol{\phi}$ and $\boldsymbol{\theta}$) using SGD.
- We can start out with random initial values of $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$, and stochastically optimize their values until convergence.
- Given a dataset with i.i.d. data, the ELBO objective is the sum (or average) of individual-datapoint ELBOs:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$$

- Unbiased gradients of the ELBO w.r.t. the generative model parameters are simple to obtain:

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\theta(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))\right] \\ &\simeq \nabla_\theta(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})) \\ &= \nabla_\theta(\log p_\theta(\mathbf{x},\mathbf{z}))\end{aligned}$$

- Unbiased gradients w.r.t. the variational parameters are more difficult to obtain, since the ELBOs expectation is taken w.r.t. the distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, which is a function of $\boldsymbol{\phi}$. I.e., in general:

$$\begin{aligned}\nabla_\phi \mathcal{L}_{\theta,\phi}(\mathbf{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right] \\ &\neq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\nabla_\phi(\log p_\theta(\mathbf{x},\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))\right]\end{aligned}$$

Reparameterization trick

Change of variables

- For continuous latent variables and a differentiable encoder and generative model, the ELBO can be straightforwardly differentiated w.r.t. both $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ through a **change of variables**, also called the **reparameterization trick**
- First, we express the random variable $\boldsymbol{z}$ as some differentiable (and invertible) transformation of another random variable $\boldsymbol{\epsilon}$, given $\boldsymbol{z}$ and

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x})$$

- where the distribution of random variable $\boldsymbol{\epsilon}$ is independent of $\boldsymbol{x}$ or $\boldsymbol{\phi}$

Gradient of expectation under change of variable

- Given such a change of variable, expectations can be rewritten in terms of $\boldsymbol{\epsilon}$

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[f(\mathbf{z})\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(\mathbf{z})\right]$$

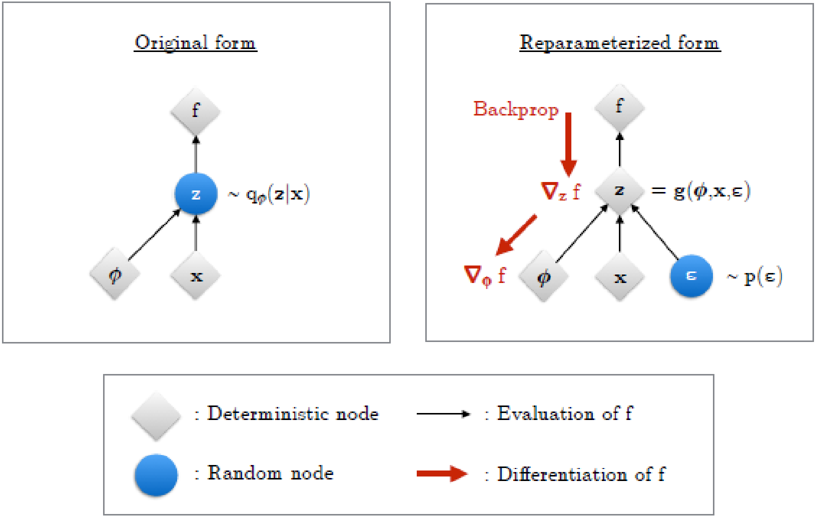
- where $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x})$ and the expectation and gradient operators become commutative, and we can form a simple Monte Carlo estimator:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[f(\mathbf{z})\right] &= \nabla_{\boldsymbol{\phi}}\mathbb{E}_{p(\boldsymbol{\epsilon})}\left[f(\mathbf{z})\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\nabla_{\boldsymbol{\phi}}f(\mathbf{z})\right] \\ &\simeq \nabla_{\boldsymbol{\phi}}f(\mathbf{z})\end{aligned}$$

- where in the last line, $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \boldsymbol{x})$ with random noise sample

$$\boldsymbol{\epsilon} \sim \boldsymbol{p}(\boldsymbol{\epsilon})$$

Illustration of the reparameterization trick



Gradient of ELBO

- Under the reparameterization, we can replace an expectation w.r.t. $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ with one w.r.t. $\boldsymbol{p}(\boldsymbol{\epsilon})$. The ELBO can be rewritten as:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]\end{aligned}$$

where $\mathbf{z} = g(\boldsymbol{\epsilon},\boldsymbol{\phi},\mathbf{x})$.

- As a result we can form a simple Monte Carlo estimator of the individual-datapoint ELBO where we use a single noise sample $\boldsymbol{\epsilon}$ from $\boldsymbol{p}(\boldsymbol{\epsilon})$:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \\ \mathbf{z} &= \mathbf{g}(\boldsymbol{\phi},\mathbf{x},\boldsymbol{\epsilon}) \\ \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\end{aligned}$$

Auto-Encoding Variational Bayes (AEVB) algorithm

Algorithm 1: Stochastic optimization of the ELBO. Since noise originates from both the minibatch sampling and sampling of $p(\boldsymbol{\epsilon})$, this is a doubly stochastic optimization procedure. We also refer to this procedure as the *Auto-Encoding Variational Bayes* (AEVB) algorithm.

Data:
  $\mathcal{D}$: Dataset
  $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model
  $p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})$: Generative model
Result:
  $\boldsymbol{\theta}, \boldsymbol{\phi}$: Learned parameters

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize parameters
while *SGD not converged* **do**
  $\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)
  $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)
  Compute $\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$
  Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer
end

Computation of $log q_\phi(\boldsymbol{z}|\boldsymbol{x})$

- Note that we typically know the density $\boldsymbol{p}(\boldsymbol{\epsilon})$, since this is the density of the chosen noise distribution. As long as $\boldsymbol{g}(.)$ is an invertible function, the densities of e and z are related as:

$$\log q_\phi(\mathbf{z}|\mathbf{x}) = \log p(\epsilon) - \log d_\phi(\mathbf{x}, \epsilon)$$

where

$$\log d_\phi(\mathbf{x}, \epsilon) = \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \epsilon} \right) \right|$$

- The Jacobian matrix contains all first derivatives of the transformation from $\boldsymbol{\epsilon}$ to $\boldsymbol{z}$:

$$\frac{\partial \mathbf{z}}{\partial \epsilon} = \frac{\partial(z_1, ..., z_k)}{\partial(\epsilon_1, ..., \epsilon_k)} = \begin{pmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \cdots & \frac{\partial z_1}{\partial \epsilon_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_k}{\partial \epsilon_1} & \cdots & \frac{\partial z_k}{\partial \epsilon_k} \end{pmatrix}$$

- we can build very exible transformations $g()$ for which $logd_{\phi}(\boldsymbol{x},\boldsymbol{\epsilon})$ is simple to compute, resulting in highly exible inference models $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$.

Factorized gaussian posteriors

A common choice is a simple factorized Gaussian encoder $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \operatorname{diag}(\boldsymbol{\sigma}^2))$:

$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_\phi(\mathbf{x})$$
$$q_\phi(\mathbf{z}|\mathbf{x}) = \prod_i q_\phi(z_i|\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2)$$

where $\mathcal{N}(z_i; \mu_i, \sigma_i^2)$ is the PDF of the univariate Gaussian distribution.

- After reparameterization, we can write:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$
$$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}) = \text{EncoderNeuralNet}_\phi(\mathbf{x})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

- The Jacobian of the transformation from $\boldsymbol{\epsilon}$ to $\boldsymbol{z}$ is:

$$\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \mathrm{diag}(\boldsymbol{\sigma}),$$

i.e. a diagonal matrix with the elements of $diag(\boldsymbol{\sigma})$ on the diagonal. The determinant of a diagonal (or more generally, triangular) matrix is the product of its diagonal terms. The log determinant of the Jacobian is therefore:

$$\log d_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}) = \log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right) \right| = \sum_i \log \sigma_i$$

and the posterior density is:

$$\begin{aligned} \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \log p(\boldsymbol{\epsilon}) - \log d_{\boldsymbol{\phi}}(\mathbf{x}, \boldsymbol{\epsilon}) \\ &= \sum_i \log \mathcal{N}(\epsilon_i; 0, 1) - \log \sigma_i \end{aligned}$$

when $\mathbf{z} = \mathbf{g}(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$.

Full-covariance Gaussian posterior

- The factorized Gaussian posterior can be extended to a Gaussian with full covariance:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

A reparameterization of this distribution is given by:

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}$$

- where ***L*** is a lower (or upper) triangular matrix, with non-zero entries on the diagonal.

The reason for this parameterization of the full-covariance Gaussian, is that the Jacobian determinant is remarkably simple. The Jacobian in this case is trivial: $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \mathbf{L}$. Note that the determinant of a triangular matrix is the product of its diagonal elements. Therefore, in this parameterization:

$$\log|\det(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}})| = \sum_i \log|L_{ii}|$$

And the log-density of the posterior is:

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \sum_i \log|L_{ii}|$$

This parameterization corresponds to the Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$ of the covariance of $\mathbf{z}$:

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathbb{E}\left[(\mathbf{z} - \mathbb{E}[\mathbf{z}|])(\mathbf{z} - \mathbb{E}[\mathbf{z}|])^T\right] \\ &= \mathbb{E}\left[\mathbf{L}\boldsymbol{\epsilon}(\mathbf{L}\boldsymbol{\epsilon})^T\right] = \mathbf{L}\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right]\mathbf{L}^T \\ &= \mathbf{L}\mathbf{L}^T\end{aligned}$$

Note that $\mathbb{E}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right] = \mathbf{I}$ since $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

One way to build a matrix $\mathbf{L}$ with the desired properties, namely triangularity and non-zero diagonal entries, is by constructing it as follows:

$$
\begin{aligned}
(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') &\leftarrow \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x}) \\
\mathbf{L} &\leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})
\end{aligned}
$$

$\mathbf{L}_{mask}$ is a masking matrix with zeros on and above the diagonal, and ones below the diagonal. The log-determinant is identical to the factorized Gaussian case:

$$
\log \left| \det \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} \right) \right| = \sum_i \log \sigma_i
$$

VAE with a full-covariance Gaussian inference model

Algorithm 2: Computation of unbiased estimate of single-datapoint ELBO for example VAE with a full-covariance Gaussian inference model and a factorized Bernoulli generative model. $\mathbf{L}_{mask}$ is a masking matrix with zeros on and above the diagonal, and ones below the diagonal. Note that $\mathbf{L}$ must be a triangular matrix with positive entries on the diagonal.

Data:

$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\epsilon}$: a random sample from $p(\boldsymbol{\epsilon}) = \mathcal{N}(0, \mathbf{I})$
$\boldsymbol{\theta}$: Generative model parameters
$\boldsymbol{\phi}$: Inference model parameters
$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model
$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

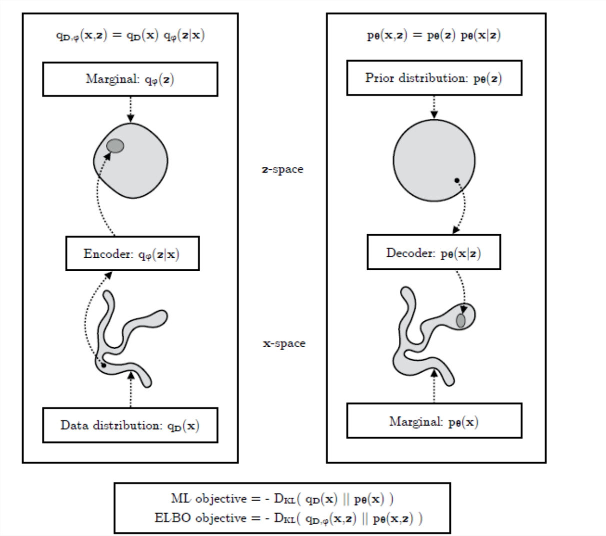
$\tilde{\mathcal{L}}$: unbiased estimate of the single-datapoint ELBO $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$

$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') \leftarrow \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x})$
$\mathbf{L} \leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
$\mathbf{z} \leftarrow \mathbf{L}\boldsymbol{\epsilon} + \boldsymbol{\mu}$
$\text{logqz} \leftarrow -\text{sum}(\frac{1}{2}(\boldsymbol{\epsilon}^2 + \log(2\pi) + \log \boldsymbol{\sigma}))$ $\quad \triangleright = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$
$\text{logpz} \leftarrow -\text{sum}(\frac{1}{2}(\mathbf{z}^2 + \log(2\pi)))$ $\quad \triangleright = p_{\boldsymbol{\theta}}(\mathbf{z})$
$\mathbf{p} \leftarrow \text{DecoderNeuralNet}_{\boldsymbol{\theta}}(\mathbf{z})$
$\text{logpx} \leftarrow \text{sum}(\mathbf{x} \odot \log \mathbf{p} + (1 - \mathbf{x}) \odot \log(1 - \mathbf{p}))$ $\quad \triangleright = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
$\tilde{\mathcal{L}} = \text{logpx} + \text{logpz} - \text{logqz}$

Estimation of the marginal likelihood

- After training a VAE, we can estimate the probability of data under the model using an importance sampling technique **Rezende et al. [2014]**

Marginal likelihood and ELBO as KL divergences



Marginal likelihood and ELBO as KL divergences

- One additional perspective is that the ELBOcan be viewed as a maximum likelihood objective in an augmented space.
- For some fixed choice of encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, we can view the joint distribution $p_\theta(\boldsymbol{x},\boldsymbol{z})$ as an augmented empirical distribution over the original data $\mathbf{x}$ and (stochastic) auxiliary features $\mathbf{z}$ associated with each datapoint.
- The model $p_\theta(\boldsymbol{x},\boldsymbol{z})$ then defines a joint model over the original data, and the auxiliary features.

Other Examples of Variational treatment

- Variational Mixture of Gaussians
- Variational Linear Regression
- Variational Logistic Regression etc.
  Bishop et al. [2006]

Thanks you.

Questions?

References

- Kingma, D.P. Variational Inference and Deep Learning: A New Synthesis (Ph.D. Thesis) (2017).
- Christopher M Bishop. Pattern recognition and machine learning, (2006).